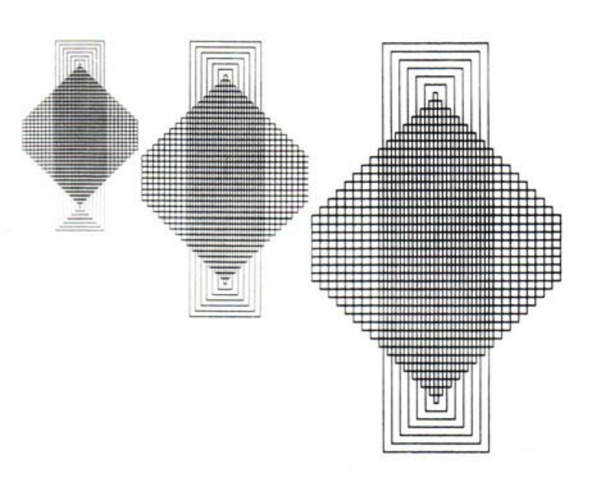
ATARI

ST BASIC™

Kurzhandbuch

Für den erfahrenen Programmierer





Inhaltsverzeichnis

Willkommen beim erweiterten ST BASIC™

Diese erweiterte Version des ST BASIC™ ersetzt das ursprüngliche ST BASIC, das mit allen ST™ Computern ausgeliefert wurde. Beide Versionen sind den Standard BASIC Dialekten ähnlich, benutzen jedoch die Fenstertechnik, die Pull-down-Menues und die graphischen Elemente des GEM™-Desktop. Außerdem nutzen sie die Geschwindigkeit und graphischen Fähigkeiten des ST-Computersystems aus.

Das neue ST BASIC läuft etwa dreimal schneller als die ursprüngliche Version und führt mehr Funktionen aus - es hat 33 neue Schlüsselworte, einen erweiterten Bereich für Integerzahlen und eine effizientere Syntax. Die Liste der Fehlermeldungen wurde erweitert und diese geben nun bei jedem Auftreten eine Klartext-Erklärung.

Das erweiterte ST BASIC ist kompatibel mit der ursprünglichen Ausgabe des ST BASIC, sodaß ältere Programme mit dieser erweiterten Version der Sprache benutzt werden können. Schlagen Sie im Abschnitt **Konvertierung von Programmen auf ST BASIC** nach, um Informationen darüber zu erhalten, wie Sie Ihre alten Programme mit der neuen Version der Sprache verwenden können.

Wie benutzt man dieses Handbuch

Dieses Handbuch ist für fortgeschrittene Programmierer ausgelegt, die die Sprache BASIC verstehen und mit den Standardprozeduren des GEM Desktop vertraut sind. Das Kurzhandbuch ist so gegliedert, daß der Programmierer die Unterschiede zwischen diesem BASIC und der vorhergehenden Version des ST BASIC verstehen kann. Darüberhinaus werden die besonderen Merkmale des BASIC auf dem ST Computer vorgestellt und demonstriert, wie Programme in anderen BASIC Dialekten geladen und mit dem erweiterten ST BASIC gestartet werden können.

Was befindet sich auf der ST Sprachdiskette

Die ST Sprachendiskette, die mit Ihrem ST Computer mitgeliefert wird, enthält die Dateien, die nötig sind, um das erweiterte ST BASIC laufen zu lassen.

BASIC.PRG ist das Programm BASIC.
BASIC.RSC ist die Resourcendatei der Sprache.

Anmerkung: Beachten Sie andere Dateien, die sich eventuell auf der Diskette befinden, nicht. Diese können ST Desktop Accessory-Programme oder Anwendungsprogramme enthalten. Ausschließlich die beiden oben genannten Dateien sind zur Programmierung mit dem erweiterten ST BASIC notwendig.

ST BASIC Handbuch

Das *ST BASIC Handbuch* ist ein komplettes Handbuch für das ST BASIC. Dieses neue Handbuch ist über 300 Seiten lang und ermöglicht einen leichten Zugriff auf alle Informationsebenen bei der Programmierung. Der Anfänger erhält einen umfangreichen Lehrgang, der erfahrene Programmierer eine komplette technische Dokumentation.

Wenn Sie an der Programmierung mit dem erweiterten ST BASIC interessiert sind, fragen Sie bitte Ihren Atarihändler nach Informationen zum *ST BASIC Handbuch.*

Wie man anfängt

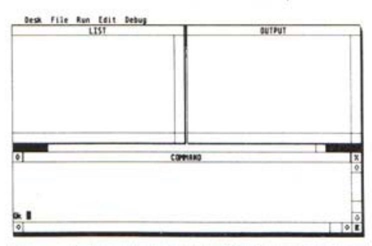
Bevor Sie mit dem ST BASIC beginnen, machen Sie bitte eine Sicherungskopie der ST Sprachendiskette. Das bewahrt Sie vor zufälliger Beschädigung auf der Original-Diskette. (Für weitere Informationen siehe das *ST Bedienungshandbuch).*

Nach der Sicherung der ST Sprachendiskette sind Sie bereit, das ST BASIC zu benutzen. Beginnen Sie mit dem Laden der Sprache auf Ihrem ST Computer, indem Sie die nachfolgenden Anweisungen ausführen.

Das Laden des ST BASIC

1. Wenn Ihr ST Computer eingeschaltet ist und das GEM Desktop angezeigt wird, klicken Sie zweimal auf dem Symbol der Diskettenstation A.

2. Wenn das Disketten-Verzeichnis erscheint, klicken Sie zweimal auf BASIC.PRG. Das ST BASIC Desktop erscheint.



Dieses Desktop ist Ihre ST BASIC Programmierumgebung.

Anmerkung: ST BASIC verwendet die Standard-Operationen des GEM Desktop, um Menuepunkte oder Optionen auszuwählen, Fenster zu manipulieren und Anwendungen zu laden. Diese Vorgehensweisen sind ausführlich im Atari *ST Bedienungshandbuch* beschrieben.

Die Konvertierung von Programmen in ST BASIC

Dieser Abschnitt beschreibt die Verbesserungen am ST BASIC, die das erweiterte ST BASIC enthält, sowie die Hauptunterschiede zwischen dem ST BASIC und anderen BASIC-Dialekten.

Das Einlesen eines BASIC-Programms

ST BASIC liest ausschließlich Programme, die als ASCII-Dateien gespeichert worden sind. Stellen Sie sicher, daß Programme, die Sie konvertieren wollen, als ASCII-Dateien vorliegen.

Eine ST BASIC Befehlszeile muß mit einer Zeilennummer beginnen, mit einem Zeilenvorschub enden und darf nicht länger als 255 Zeichen sein. Gültige Zeilennummern reichen von 1 bis 65529. ST BASIC erkennt keine Zeilenfortsetzungszeichen (in einigen BASICs der Zeilenvorschub); der zweite Teil jeder Zeile, die in solcher Weise fortgesetzt ist, geht verloren.

Mit den obigen Ausnahmen erhält das ST BASIC den Text Ihres Programms, selbst wenn es Syntaxfehler meldet; Sie können den ST BASIC Editor benutzen, um das Programm zu ändern. Der einfachste Weg, ein BASIC-Programm zu konvertieren, ist, es in das ST BASIC zu laden, zu prüfen, welche Fehler gemeldet werden und diese Zeilen direkt zu editieren.

Bestehende Unterschiede

Die folgenden Informationen diskutieren die Unterschiede zwischen dem erweiterten ST BASIC und anderen BASICs.

ST BASIC

Das erweiterte ST BASIC ist kompatibel mit der früheren Ausführung des ST BASIC. Programme, die in der früheren Version geschrieben wurden, können mit dieser neuen benutzt werden, wenn die folgenden Unterschiede berücksichtigt werden.

DEF SEG wurde gestrichen. Stattdessen gibt es besondere Versionen von PEEK und POKE für Worte, Bytes und Langworte. Diese sind PEEK_W, PEEK_B, PEEK_L und POKE_W, POKE_B und POKE_L. Programme, die DEF SEG im alten ST BASIC benutzen, müssen überarbeitet werden.

Die Adressen für PEEK und POKE benutzen Integerzahlen, sodaß die Genauigkeit ausreichend ist.

Integerzahlen sind jetzt 32-Bit-Zahlen. Dadurch wird deren gültiger Bereich erweitert, der bisher von -32768 bis 32767 ging. Der Bereich liegt jetzt zwischen -2147483648 und 2147483647.

Es wurde eine neue Syntax für GEMSYS und VDISYS eingeführt, die effizienter arbeitet als die alte. Diese arbeitet noch korrekt mit einem Zusatz: die Zahl in Klammern muß in GEM_CONTROL(0) für GEMSYS oder in CONTRL(0) für VDISYS abgelegt werden. Programme, die VDISYS und GEMSYS benutzen, sollten an die neue Syntax angepaßt werden.

Neue reservierte Worte wurden hinzugefügt. Dies sind:

AREA	GEM_ADDROUT	PATTERN
ASK MOUSE	GEM_CONTROL	PEEK_B
ASK RGB	GEM_GLOBAL	PEEK_L
BIOS	GEM_INTIN	PEEK_W
BOX	GEM_INOUT	POKE_B
CLEAR	GEMDOS	POKE_L
DRAW	GSHAPE	POKE_W
DRAWMODE	LINEPAT	RGB
ED	MAT AREA	SSHAPE
ERR$	MAT DRAW	STATUS
GEM_ADDRIN	MAT SOUND	XBIOS

Programme, die mit dem früheren ST BASIC geschrieben wurden und irgendeines dieser Worte anders als als reserviertes Wort benutzen, müssen überarbeitet werden.

Jede Syntax, die eine Liste von x,y-Paaren benötigt, wird in der Dokumentation mit einem Semikolon (;) zwischen den Paaren beschrieben. Das Semikolon fördert die Lesbarkeit, die alte Syntax funktioniert aber nach wie vor.

SYSTAB ist jetzt ein Array von 2-Byte Integerzahlen. Zugriffe auf SYSTAB sind jetzt solche auf Arrayelemente, deren Index die Hälfte des vorherigen Offset ist: z.B. SYSTAB + 6 wird zu SYSTAB(3).

INP im Zusammenhang mit -1 wird nicht immer eine negative Zahl ergeben; allerdings ist das Ergebnis immer ungleich Null.

Der Punkt (.) darf nicht mehr in Variablennamen und Schlüsselwörtern auftreten und sollte durch einen Unterstrich ersetzt werden.

Andere BASIC-Dialekte

Identifikatoren

Variablennamen und Schlüsselwörter müssen im ST BASIC mit einem Buchstaben anfangen und können Buchstaben (A bis Z oder a bis z), Ziffern (0 bis 9) und den Unterstrich (_) enthalten. Beachten Sie, daß große und kleine Buchstaben gleichwertig sind. Andere BASICs erlauben den Punkt (.), aber nicht den Unterstrich, und in einigen Fällen ist die Groß-/Kleinschreibung von Bedeutung.

Stringkonstanten
ST BASIC ermöglicht es Ihnen, Anführungsstriche (") in einer Stringkonstanten zu benutzen, wenn sie von einem weiteren Paar Anführungsstriche umschlossen sind. Zum Beispiel:

```
PRINT ""Das ist wirklich eine Zitrone.", sagte er."
```

wird gedruckt als

```
"Das ist wirklich eine Zitrone.", sagte er.
```

Andere BASIC-Dialekte behandeln das Anführungszeichen als das Ende einer Stringkonstante und Anfang einer nächsten. Da Trennzeichen zwischen den einzelnen Elementen einer PRINT-Liste nicht obligatorisch sind, kann es vorkommen, daß ein Befehl wie

```
PRINT "A""B"
```

eines existierenden Programms im ST BASIC nicht das tut, was Sie erwarten. ST BASIC druckt "A" "B" als A"B; andere BASICs drucken es als AB.

Arithmetik
ST BASIC kennt drei numerische Zahlenarten: Integerzahlen und Fließkommazahlen einfacher und doppelter Genauigkeit.

Integerzahlen belegen 4 Bytes und können Zahlen von -2147483648 bis 2147483647 darstellen. In vielen anderen BASIC-Dialekten belegen Integerzahlen 2 Bytes und repräsentieren Zahlen im Bereich von -32768 bis 32767. Dieser Sachverhalt hat Auswirkungen sowohl auf die Befehle MKI$ und CVI als auch auf das Format von Datensätzen, die Integerzahlen enthalten.

ST BASIC wertet Ausdrücke, die ausschließlich Integerterme enthalten, rein als Integer aus, was bedeuten kann, daß Zwischenergebnisse überlaufen können. Zum Beispiel:

```
a%=b%*c%-170000
```

wird als Integer ausgewertet.

Fließkommawerte werden im IEEE-Format behandelt. Dieses weicht geringfügig von einigen anderen Formaten ab in Bezug auf den dynamischen Bereich und die Präzision. Das ST BASIC erkennt keine nichtnormalisierten Zahlen, NANs oder unendliche Werte. Zum Beispiel:

```
PRINT 1E-38
```

wird als 0 gedruckt, da 1E-38 kleiner ist als die kleinste, normalisierte Zahl einfacher Genauigkeit im IEEE-Format.

Arithmetik mit gemischten Darstellungsformen wird in doppelter Genauigkeit ausgeführt, da die Genauigkeit der Integerzahlen (31 bits plus Vorzeichen) größer ist als die der Fließkommazahlen einfacher Genauigkeit (24 bits plus Vorzeichen).

Übersetzung und Interpretation
ST BASIC übersetzt ein Programm in internen Code entweder direkt wenn Sie es eingeben oder wenn Sie es über LOAD laden. Wenn Sie ein Programm mit RUN aufrufen, wird der interne Code ausgeführt. Dies ist ein Unterschied zu den meisten BASICs für Mikrocomputer, die nach dem Aufruf fortlaufend übersetzen und ausführen, wodurch sich definierende Befehle unterschiedlich auswirken.

DEF-Art Befehle (die Default-Variablenarten definieren) verändern die Arbeitsweise des Übersetzers. Sie werden wirksam, sobald sie eingegeben werden, unabhängig davon, wo sie im Programm stehen. Deshalb ist das folgende Beispiel in ST BASIC nicht möglich:

```
100 input "Welche Art";a%
110 if a%=1 then gosub 2000: goto 500
130 a=5.0: goto 600
500 a="ja"
600 print a
700 end
2000 defstr a
2010 return
```

DEF FN definiert eine Benutzerfunktion, unabhängig davon, wo es im Programm steht. Die DEF FN-Funktion muß nicht erst ausgeführt werden, um die Funktion zu definieren. Zwei DEF FN-Befehle, die dieselbe Funktion definieren, sind nicht möglich. Deshalb ist das folgende Beispiel im ST BASIC nicht möglich:

```
100 input "Welches def";a%
110 if a%=1 then gosub 2000 else gosub 2100
120 print FNR(1,2)
130 end
2000 def fnr(x,y)=x/y
2010 return
2100 def fnr(x,y)=y/x
2110 return
```

FOR/NEXT Einschränkungen
ST BASIC fordert für jeden FOR-Befehl genau einen NEXT-Befehl und für jeden WHILE-Befehl genau einen WEND-Befehl. Der Abgleich wird durchgeführt, bevor das Programm ausgeführt wird. Viele BASICs führen ihn erst während der Ausführung durch, sodaß die folgende Konstruktion möglich ist:

```
100 FOR i%=1 TO 1000
   .
   .
   .
300 IF i%>500 THEN NEXT
   .
   .
   .
500 NEXT
```

ST BASIC meldet hier einen Fehler in Zeile 300, da es vor der Programmausführung nicht entscheiden kann, ob das NEXT in der Zeile 300 zu dem FOR aus der Zeile 100 gehört oder nicht.

ST BASIC erlaubt keine Sprünge von außen in FOR/NEXT oder WHILE/WEND Schleifen. Für das Beispiel

```
10 GOTO 200
100 FOR i%=1 TO 1000
   .
   .
   .
200 PRINT "Der Wert von i% ist ";i%
   .
   .
   .
300 NEXT
```

meldet ST BASIC einen Fehler, wenn das Programm aufgerufen wird. Andere BASICs führen das Programm evtl. aus und melden einen Fehler wie etwa "NEXT ohne FOR" in der Zeile 300.

Umgebung der Kommandos
ST BASIC unterscheidet zwischen Befehlen (z.B. PRINT), die Teil eines Programms sind, dieses aber nicht verändern können, und Kommandos, die benutzt werden, um ein Programm zu überprüfen oder zu verändern, die selbst nicht Teil eines Programms sein können. Einige andere BASICs lassen es zu, daß Kommandos selbst Teil des Programms sind, auf das sie sich auswirken. ST BASIC meldet einen Fehler, wenn es ein Kommando in einer (mit Zeilennummer versehenen) Programmzeile findet. Die ST BASIC-Kommandos sind auf Seite 12 aufgelistet.

Fehlernummern und Meldungen

Die folgenden Fehlernummern und Fehlermeldungen werden bei ERL, ERR, ERR$ und ERROR benutzt. (Die Erklärung dieser reservierten Wörter finden Sie im Abschnitt **Kommandos, Befehle, Funktionen und Systemvariablen**.)

Nummer	Meldung
0-1	Nicht definiert
2	Syntaxfehler
3	RETURN ohne GOSUB
4	Keine Daten mehr
5	Ungültiger Funkt.aufruf
6	Überlauf
7	Kein Speicher mehr
8	Undef. Zeilennummer
9	Ungültiger Index
10	Doppelte Definition
11	Division durch Null
12	Ungültig im Direktmodus
13	Unterschiedl. Arten
14	Nicht definiert
15	String zu lang
16	Ausdruck zu komplex
17	CONT nur im BREAK-Modus möglich
18	Undefinierte Benutzerfunktion
19	Nicht definiert
20	RESUME ohne Fehler
21	Nicht definiert
22	Fehlender Operand
23	Programmzeile zu lang
24-49	Nicht definiert
50	Feld-Überlauf
51	Ungültige Satzlänge
52	Ungültige Dateinummer
53	Datei nicht gefunden
54	Ungültiger Dateimodus
55	Datei bereits geöffnet
56	Nicht definiert
57	Geräte I/O-Fehler
58	Datei existiert
59	Datei kann nicht angelegt werden
60	Nicht definiert
61	Diskette ist voll
62	Ende der Datei
63	Ungültige Satznummer
64	Ungültiger Dateiname
65	Ungültiges Zeichen in der Programmdatei
66	Direkter Befehl in der Datei
67	Löschung schlug fehl
68	Gerät nicht verfügbar
69-92	Nicht definiert
93	Nicht definiertes Segment
94	Geschützte Datei
95	Kein BASIC-Programm
96-98	Nicht definiert
99	Break
100	Nicht definiert
101	Programm ist zu groß
102	Nicht definiert
103	Ungültige Zeilennummer
104	Fehlende Zeilennummer
105	Nicht definiert
106	Befehl nicht gefunden
107	Integer-Überlauf
108	Nochmal vom Anfang an
109	Stop
110	Zu tiefe GOSUB-Verschachtelung
111	Ungültige BLOAD-Datei
112-200	Nicht definiert
201	Ungültige Option
202	Kommando hier nicht erlaubt
203	Zeilennummer erforderlich
204	FOR ohne NEXT
205	NEXT ohne FOR
206	Komma fehlt
207	Klammer fehlt
208	Optionaler Anfang muß 0 oder 1 sein
209	Nicht definiert
210	WHILE ohne WEND
211	WEND ohne WHILE
212	Nicht definiert
213	Doppeltes DEF FN
214	Ungültiger Sprung in eine Schleife
215	Doppelte Zeilennummer
216	Doppelte Marke
217-220	Nicht definiert
221	Systemfehler #%u
222	Programm nicht ausgeführt
223	Zu viele FOR-Schleifen
224-230	Nicht definiert

Kurzhandbuch

Definierende Zeichen

Zeichen	Funktion
%	Definiert Zeichen als Integerzahl
$	Definiert Zeichen als String
!	Definiert Zeichen als Zahl einfacher Genauigkeit
#	Definiert Zeichen als Zahl doppelter Genauigkeit

Trennzeichen

Zeichen	Aufgabe
'	Begrenzt Kommentare
"	Begrenzt Strings
;	Begrenzt Eingabeprompts
,	Trennt Argumente voneinander
:	Trennt Befehle voneinander

Kommandos

Funktionstaste	Option
[F1]	Füge Leerzeichen ein
[F2]	Lösche Zeichen
[F3]	Füge neue Zeile ein
[F4]	Lösche Zeile
[F5]	Vorblättern
[F6]	Zurückblättern
[F7]	Text laden
[F8]	Text sichern
[F9]	Speicher leeren
[F10]	Verlassen des Editierungsfensters
EDIT[Return]	Starte Editierung

Operatoren

Arithmetische	Operatoren Aufgabe
+	Addiert; fügt Strings zusammen
–	Subtrahiert; negiert
*	Multipliziert
/	Dividiert
\	Verwandelt in Integerzahl und dividiert
^ oder **	Potenziert

Arithmetische Operatoren	Aufgabe
=	Ist gleich
<	Ist kleiner
>	Ist größer
<=	Ist kleiner oder gleich
>=	Ist größer oder gleich
><	Ist ungleich

Logische Operatoren	Aufgabe
AND	Führt das logische "und" aus
EQV	Prüft auf logische Gleichheit
IMP	Prüft auf logische Implikation
NOT	Negiert den Ausdruck
OR	Führt das logische "oder" aus
XOR	Führt das logische "exklusiv oder" aus

Zeichenmodus

Nummer	Modus
1	Ersetzen
2	Transparent
3	Exklusiv ODER
4	Umgekehrt transparent

Füllmuster

Nummer	Art
0	Hohl
1	Durchgehend
2	Muster
3	Schraffiert

Linienmuster

Nummer	Art
1	Durchgehend
2	Lange Striche
3	Punkte
4	Strich-Punkt
5	Striche
6	Strich-Punkt-Punkt
7	Benutzer definiert

Datenkanäle

Nummer	Datenkanal
0	Drucker
1	Modem
2	Konsole
3	MIDI

Fenster

Nummer	Fenster
0	Editierung
1	Programmliste
2	Programmausgabe
3	Befehlseingabe

Ton

Parameter	Beschreibung	Bereich
Dauer	1/50 Sek. Zeit vor dem nächsten Ton	
Note	Regelt den Klang: Lage der Note in der Tonleiter	1 bis 12
Oktave	Regelt den Klang: Nummer der Oktave	1 bis 8
Stimme	Nummer des Tonkanals	1 bis 3
Lautstärke	Regelt die Lautstärke	0(aus) bis 15(max.)

Liste der Kommandos

AUTO
BREAK
CONT
DELETE
DIR
EDIT
ERA
FOLLOW
LIST
LLIST
LOAD
MERGE
NEW
RENUM
REPLACE
RUN
SAVE
STEP
TRACE
TROFF
TRO
UNBREAK
UNFOLLOW
UNTRACE

Liste der Befehle

AREA
ASK MOUSE
ASK RGB
BLOAD
BOX
BSAVE
CALL
CHAIN(MERGE)
CIRCLE
CLEAR
CLEARW
CLOSE
CLOSEW
COLOR
COMMON
DATA
DEF FN
DEFDBL
DEFINT
DEFSNG
DEFSTR
DIM
DRAW
DRAWMODE
ELLIPSE
END
ERASE
ERROR
FIELD
FILL
FOR
FULLW
GEMDOS
GET
GOSUB
GOTO
GOTOXY
GSHAPE
IF
INPUT[#]
KILL
LET
LINE INPUT[#]
LINEF
LINEPAT
LPRINT
LSET
MAT AREA
MAT DRAW
MAT LINEF
MAT SOUND
NAME
NEXT
ON
ON ERROR GOTO
OPEN
OPENW
OPTION BASE
OUT
PATTERN
PCIRCLE
PELLIPSE
PRINT[#]
PRINT USING
PUT
QUIT
RANDOMIZE
READ
REM
RESET
RESTORE
RESUME
RETURN
RGB
RSET
SOUND
SSHAPE
STOP
SWAP
SYSTEM
WAVE
WEND
WHILE
WIDTH
WRITE[#]
XBIOS

Liste der Funktionen

ABS
ASC
ATN
BIOS
CDBL
CHR$
CINT
COS
CSNG
CVD
CVI
CVS
EOF
ERR$
EPX
FIX
FLOAT
FRE
GEMSYS
HEX$
INP
INPUT$
INSTR
INT
LEFT$
LEN
LOC
LOF
LOG
LOG10
LPOS
MID$
MKD$
MKI$
MKS$
OCT$
PEEK
PEEK_B
PEEK_L
POKE
POKE_B
POKE_L
POS
RIGHT$
RND
SGN
SIN
SPACE$
SPC
SQR
STR$
STRING$
TAB
TAN
VAL
VARPTR
VDISYS

Liste der Systemvariablen

CONTRL
ERL
ERR
GEM_ADDRIN
GEM_ADDROUT
GEM_CONTRL
GEM_GLOBAL
GEM-INTIN
GEM_INTOUT
INTIN
INTOUT
PI
PTSIN
PTSOUT
STATUS
SYSTAB

Kommandos, Befehle, Funktionen und Systemvariablen

Name	Format/Beschreibung
=	<Variable>= <Numerischer Ausdruck> I "String" Weist einer Variablen einen Wert zu (siehe LET).
ABS	ABS(<Numerischer Ausdruck>) Gibt den Absolutbetrag des Arguments aus.
AREA	AREA<Liste von Punkten> Zeichnet ein ausgefülltes Polygon.
ASC	ASC(<Stringausdruck>) Gibt den ASCII-Code des spezifizierten Zeichens aus.
ASK MOUSE	ASK MOUSE>x<,>y<,>t< Gibt die aktuellen Koordinaten der Maus und den Tastenstatus in die angeführten Variablen aus.
ASK RGB	ASK RGB<Reg>,<r>,<g>,<b> Weist den angegebenen Variablen die aktuellen Rot-, Grün- und Blauwerte der spezifizierten Farbpalette zu.
ATN	ATN(<Numerischer Ausdruck>) Gibt den Arcustangens des Arguments aus.
AUTO	AUTO[<Startzeile>][,<Erhöhung>] Nummeriert bei der Eingabe die Zeilen automatisch.
BIOS	BIOS<Numerischer Ausdruck>,< Liste der Argumente> Erlaubt einen Betriebssystemaufruf des BIOS.
BLOAD	BLOAD<Dateiangabe>,<Adresse> Lädt die angegebene, binäre Datei.
BOX	BOX[FILL]<x1,y1>;<x2,y2> Zeichnet ein Rechteck.
BREAK	BREAK[<Liste von Zeilennummern>] Unterbricht die Programmausführung an den angegebenen Zeilen.

BSAVE BSAVE<Dateiangabe>,<Adresse>,<Länge>
Speichert einen Teil des Speichers in eine binäre Datei.

CALL CALL<Numerische Variable>[(<Parameterliste>)]
Ruft ein Unterprogramm in Maschinensprache auf.

CDBL CDBL(<Numerischer Ausdruck>)
Wandelt das Argument in eine Zahl doppelter Genauigkeit um.

CHAIN CHAIN<Dateiangabe>[,<Zeilenbezeichnung>][,ALL]
Ersetzt das gegenwärtige Programm durch das angegebene und führt dieses aus.

CHAIN MERGE CHAIN MERGE<Dateiangabe>[,<Zeilenbezeichnung>]
[,DELETE <Zeilenbezeichnung>]
Fügt das angegebene Programm in das gegenwärtige ein und führt das daraus resultierende Programm aus.

**CHR$** CHR$(<Numerischer Ausdruck>)
Wandelt Integerzahlen entsprechend dem ASCII-Code in Strings der Länge 1 um.

CINT CINT(<Numerischer Ausdruck>)
Wandelt das Argument in eine Integerzahl um.

CIRCLE CIRCLE<x>,<y>,<Radius> [,<Anfangswinkel,Endwinkel>]
Zeichnet Kreise und Kreissegmente.

CLEAR CLEAR
Setzt alle numerischen Variablen gleich 0 und alle Stringvariablen gleich dem Nullstring. Die Dimensionierung aller Arrays wird rückgängig gemacht.

CLEARW CLEARW<Numerischer Ausdruck>
Löscht ST BASIC-Bildschirmfenster.

CLOSE CLOSE[#]<Dateinummer>
Beendet Ein- und Ausgabe zu einer Datendatei und schließt sie.

CLOSEW CLOSEW<Nummer des Fensters>
Schließt BASIC-Fenster.

COLOR COLOR[<Textfarbe,Füllfarbe,Linienfarbe,Index,Art>]
Setzt die Textfarbe, Füllfarbe und Zeichenfarbe sowie das Füllmuster.

COMMON COMMON<Variable>[,<Variable>...]
Übergibt die angegebenen Variablen an ein mit CHAIN aufgerufenes Programm.

CONT CONT
Nimmt die Programmausführung nach einer Unterbrechung durch BREAK wieder auf.

CONTRL CONTRL(<Offset>)=<Ausdruck>
Eine Systemvariable, die mit der Handhabung des VDI verbunden ist und als Array im Programm benutzt werden kann.

COS COS(<Numerischer Ausdruck>)
Gibt den Kosinus des Arguments aus.

CSNG CSNG(<Numerischer Ausdruck>)
Wandelt das Argument in eine Zahl einfacher Genauigkeit um.

CVD CVD(<8-Byte String>)
Wandelt einen 8 Byte langen String in eine Zahl doppelter Genauigkeit um.

CVI CVI(<4-Byte String>)
Wandelt einen 4 Byte langen String in eine Integerzahl um.

CVS CVS(<4-Byte String>)
Wandelt einen 4 Byte langen String in eine Zahl einfacher Genauigkeit um.

DATA DATA<Konstante>[,<Konstante>...]
Stellt die Daten dar, die über einen READ-Befehl gelesen werden.

DEF FN DEF FN<Funktionsname>[(<Variablenliste>)]=<Definition>
Definiert eine Benutzerfunktion.

DEFDBL DEFDBL<Buchstabe>[-<Buchstabe>]
Definiert einen Bereich von Anfangsbuchstaben als Zahlen doppelter Genauigkeit.

DEFINT DEFINT<Buchstabe>[-<Buchstabe>]
Definiert einen Bereich von Anfangsbuchstaben als Integerzahlen.

DEFSNG	DEFSNG<Buchstabe>[-<Buchstabe>] Definiert einen Bereich von Anfangsbuchstaben als Zahlen einfacher Genauigkeit.
DEFSTR	DEFSTR<Buchstabe>[-<Buchstabe>] Definiert einen Bereich von Anfangsbuchstaben als String.
DELETE	DELETE<Zeilennummer>[-<Zeilennummer>] Löscht Programmzeilen aus dem Speicher.
DIM	DIM<Arrayname>(<Index>[,<Index>]...) [,<Arrayname>(<Index>,<Index>...)] Verbindet den Variablennamen mit einem Array der angegebenen Dimensionen.
DIR	DIR[<Laufwerk>:] [[<Dateiname>].[<Erweiterung>]] Listet ein Inhaltsverzeichnis.
DRAW	DRAW<Punktliste> Zeichnet eine Linie durch die Punkte, die als Argumente aufgeführt sind.
DRAWMODE	DRAWMODE<Integervariable> Setzt den gültigen Zeichenmodus.
EDIT	EDIT[<Zeilennummer>] Editiert das gegenwärtige Programm.
ELLIPSE	ELLIPSE<x>,<y>,<Horizontaler Radius>, <Vertikaler Radius>[,<Anfangswinkel>,<Endwinkel>] Zeichnet eine Ellipse.
END	END Beendet das Programm, schließt die Dateien und kehrt in den Direktmodus zurück.
EOF	EOF(<Dateinummer>) Stellt das Ende der Datei fest.
ERA	ERA[<Laufwerk:>][<Dateiname>] Löscht eine Datei auf der Diskette.
ERASE	ERASE<Arrayname>[,<Arrayname>]... Löscht Arrays.
ERL	ERL=<Fehlerhafte Zeile> Enthält die Nummer der Zeile, in der ein Fehler auftrat.
ERR	ERR=<Fehlernummer> Enthält die Fehlernummer.

ERR$
ERR$(<n>)
Gibt eine Fehlermeldung für die angegebene Fehlernummer aus.

ERROR
ERROR<Numerischer Ausdruck>
Simuliert einen Fehler.

EXP
EXP(<Numerischer Ausdruck>)
Potenziert e mit dem angegebenen Exponenten.

FIELD
FIELD #<Dateinummer>,<Feldbreite>AS <Stringvariable>[,<Feldbreite>AS <Stringvariable>]...
Weist den Platz für Werte der Variablen im Dateipuffer zu.

FILL
FILL<x>,<y>
Füllt Figuren mit Farben oder Mustern.

FIX
FIX(<Numerischer Ausdruck>)
Schneidet die Nachkommastellen des Arguments ab.

FLOAT
FLOAT(<Integerausdruck>)
Wandelt eine Integerzahl in eine Zahl einfacher Genauigkeit um.

FOLLOW
FOLLOW<Variable>[,<Variable>...]
Verfolgt die Werte der angegebenen Variablen während der Programmausführung.

FOR...TO
FOR<Zählvariable>=<Anfangswert>TO<Grenzwert> [STEP<Erhöhung>]
Stellt den Anfang einer Programmschleife dar.

FRE
FRE[<Numerischer Ausdruck>]
Gibt die Anzahl der Bytes im Speicher aus, die von ST BASIC nicht benutzt werden.

FULLW
FULLW<Fensternummr>
Bringt ST BASIC-Fenster auf die volle Bildschirmgröße.

GEM_ADDRIN
GEM_ADDRIN(<Offset>)=<Ausdruck>
Eine GEM-Systemvariable, die als Array im Programm benutzt werden kann.

GEM_ADDROUT
GEM_ADDROUT(<Offset>)=<Ausdruck>
Eine GEM-Systemvariable, die als Array im Programm benutzt werden kann.

GEM_CONTRL GEM_CONTRL(<Offset>)=<Ausdruck>
Eine GEM-Systemvariable, die als Array im Programm benutzt werden kann.

GEM_GLOBAL GEM_GLOBAL(<Offset>)=<Ausdruck>
Eine GEM-Systemvariable, die als Array im Programm benutzt werden kann.

GEM_INTIN GEM_INTIN(<Offset>)=<Ausdruck>
Eine GEM-Systemvariable, die als Array im Programm benutzt werden kann.

GEM_INTOUT GEM_INTOUT(<Offset>)=<Ausdruck>
Eine GEM-Systemvariable, die als Array im Programm benutzt werden kann.

GEMDOS GEMDOS<Numerischer Ausdruck>,<Argumentenliste>
Ermöglicht einen Betriebssystemsaufruf von GEMDOS.

GEMSYS GEMSYS<AES Op Code>
Greift auf die AES-Schnittstelle des Betriebssystems zu.

GET GET[#]<Dateinummer>[,<Satznummer>]
Liest einen Satz aus einer Randomdatei in den Puffer.

GOSUB GOSUB<Zeilenbezeichnung>
Übergibt die Programmkontrolle an ein Unterprogramm.

GOTO GOTO<Zeilenbezeichnung>
Das Programm verzeweigt zur angegebenen Zeile.

GOTOXY GOTOXY<Spaltenposition>,<Reiheposition>
Plaziert den Cursor des Ausgabebereichs in der angegebenen Spalte und Reihe.

GSHAPE GSHAPE<x1>,<y1>,<Array>
Schreibt das Raster, das in dem angegebenen Array gespeichert ist, auf den Bildschirm.

**HEX$** HEX$(<Numerischer Ausdruck>)
Gibt den hexadezimalen Wert des Arguments als String aus.

IF IF <Beziehung> THEN <Zeilenzahl> I <Marke>
IF <Beziehung> GOTO <Zeilenzahl> I <Marke>
IF <Beziehung> THEN <Zeilenzahl>
[ELSE <Zeilenzahl> I <Marke>]
IF <Beziehung> THEN <Klause> [ELSE <Klause>]
Wertet die Beziehung aus und entscheidet dem Ergebnis entsprechend über den weiteren Ablauf des Programms.

INP INP(<Nummer des Datenkanal>)
Gibt als Wert ein Byte von einem ausgewählten Datenkanal zurück.

INPUT INPUT[;][<Prompttext><;l,>]
<Variable>[,<Variable>]...
Liest während der Programmausführung Eingaben von der Tastatur ein.

INPUT# INPUT#<Dateinummer>,<Variable>[,<Variable>]...
Ordnet der (den) angegebenen Variablen Daten aus einer sequentiellen Datei zu.

**INPUT$** INPUT$(<Anzahl der Zeichen>[,[#]<Dateinummer>])
Liest einen String der angegebenen Länge von der Tastatur oder der Datei ein.

INSTR INSTR([<Anfangsposition>,] <Zielstring>, <Muster-String>)
Sucht einen String in einem anderen und gibt dessen Position aus.

INT INT(<Numerischer Ausdruck>)
Rundet das Argument auf die nächst kleinere ganze Zahl ab.

INTIN INTIN(<Offset>)=<Ausdruck>
Eine VDI-Systemvariable, die als Array im Programm benutzt werden kann.

INTOUT INTOUT(<Offset>)=<Ausdruck>
Eine VDI-Systemvariable, die als Array im Programm benutzt werden kann.

KILL KILL <Stringausdruck>
Löscht eine Datei.

**LEFT$** LEFT$(<Zielstring>,<Anzahl der Zeichen>)
Gibt einen Teilstring aus, vom Anfang startend.

LEN LEN(<Stringausdruck>)
Gibt die Länge des angegebenen String aus.

LET LET<Variable>=<Numerischer Ausdruck> "String"
Weist einer Variablen einen Wert zu.

LINE INPUT LINE INPUT[[;]"Promptstring";
[;]"Promptstring",]<Stringvariable>
Weist einen Satz einer sequentiellen Datei der angegebenen Variablen zu.

LINE INPUT# — LINE INPUT#<Dateinummer>,<Stringvariable>
Liest eine Zeile eines Datenfiles und weist ihn der Stringvariablen zu.

LINEF — LINEF<Punktliste>
Zeichnet eine Linie durch die Punkte der Argumentenliste.

LINEPAT — LINEPAT<Art>[,<Muster>]
Setzt das Linienmuster.

LIST — LIST[<Liste von Zeilenbezeichnungen>]
Zeigt die angegebene(n) Programmzeile(n) im Fenster LIST.

LLIST — LLIST[<Liste von Zeilenbezeichnungen>]
Druckt die angegebene(n) Programmzeile(n).

LOAD — LOAD<Dateiname>
Lädt das angegebene Programm.

LOC — LOC(<Dateinummer> oder <Satznummer>)
Gibt die Satznummer oder die Zahl der geschriebenen bzw. gelesenen Zeichen aus.

LOF — LOF(<Dateinummer>)
Gibt die Länge der Datei aus.

LOG — LOG(<Numerischer Ausdruck>)
Berechnet den natürlichen Logarithmus des Arguments und gibt ihn aus.

LOG10 — LOG10(<Numerischer Ausdruck>)
Berechnet den Logarithmus zur Basis 10 des Arguments und gibt ihn aus.

LPOS — LPOS[(Scheinargument)]
Gibt die Position des Druckkopfs aus.

LPRINT — LPRINT[<Liste von Ausdrücken>]
Druckt die Daten.

LPRINT USING<Format-Stringausdruck>;
[<Liste von Ausdrücken>]
Druckt die Daten unter Benutzung des angegebenen Formats.

LSET — LSET<Stringvariable>=<Stringausdruck>
Bringt die Daten links ausgerichtet in die Stringvariable.

MAT	AREA MAT AREA<Zahl>,<Array> Zeichnet ein gefülltes Polygon unter Benutzung der Eckpunkte aus dem angegebenen Array.
MAT DRAW	MAT DRAW<Zahl>,<Array> Zeichnet eine Linie unter Benutzung der Eckpunkte aus dem angegebenen Array.
MAT LINEF	MAT LINEF<Zahl>,<Array> Zeichnet eine Linie unter Benutzung der Eckpunkte aus dem angegebenen Array.
MAT SOUND	MAT SOUND<Array> Übergibt das angegebene Array an den Tongenerator.
MERGE	MERGE<Dateiname> Fügt das angegebene Programm dem residenten Programm hinzu.
MID$	MID$("Zielstring",<Anfang>[,<Länge>]) Gibt den angegebenen Unterstring aus dem Zielstring aus.
MID$	MID$(<Stringvariable>,<Anfang>[,<Länge>])="String" Ersetzt einen Teilstring durch einen anderen in dem existierenden Wert der Stringvariablen.
MKD$	MKD$(<Numerischer Ausdruck>) Wandelt eine Zahl doppelter Genauigkeit in einen String um.
MKI$	MKI$(<Integerzahl>) Wandelt eine Integerzahl in einen String um.
MKS$	MKS$(<Numerischer Ausdruck>) Wandelt eine Zahl einfacher Genauigkeit in einen String um.
NAME	NAME<Alter Dateiname>AS<Neuer Dateiname> Ändert den Namen einer Datei.
NEW	NEW[<Dateiname>] Löscht den Inhalt des Speichers für ein neues Programm und gibt diesem ggf. einen Namen.
NEXT	NEXT[<Zähler>[,<Zähler>]]... Gibt das Ende einer Schleife an.
OCT$	OCT$(<Numerischer Ausdruck>) Gibt den Oktalwert des Arguments in Stringform aus.

ON
ON<Ausdruck>GOSUB<Zeilenbezeichnung>
[,<Zeilenbezeichnung>]...
Definiert eine mehrfache Verzweigung zu Unterprogrammen.

ON<Ausdruck>GOTO<Zeilenbezeichnung>
[,<Zeilenbezeichnung>]
Definiert eine mehrfache Verzweigung.

ON ERROR GOTO
ON ERROR GOTO 0<Zeilenbezeichnung>
Definiert die Anfangszeile einer Fehlerabfangroutine.

OPEN
OPEN"Modus",#<Dateinummer>,"Dateiname"
[,<Satzlänge>]
Öffnet die angegebene Datendatei.

OPENW
OPENW<Fensternummer>
Öffnet ST BASIC-Fenster.

OPTION BASE
OPTION BASE<01>
Definiert den Anfangswert für die Dimensionierung von Arrays.

OUT
OUT<Ausgangsnummer>,<Byte>
Sendet ein Byte zu dem ausgewählten Datenkanal.

PATTERN
PATTERN<Ebene>,<Array>
Setzt die Art des Füllmusters.

PCIRCLE
PCIRCLE<x>,<y>,<Radius>
[,<Anfangswinkel>,<Endwinkel>]
Zeichnet gefüllte Kreise und Kreissegmente.

PEEK
PEEK_B(<Adresse>
Gibt den 8-Bit-Wert an der Speicheradresse aus.

PEEK(<Adresse>)
Gibt den 16-Bit-Wert an der Speicheradresse aus.

PEEK_L(<Adresse>)
Gibt den 32-Bit-Wert an der Speicheradresse aus.

PELLIPSE
PELLIPSE<x>,<y>,<Horizontaler Radius>,<Vertikaler Radius>,<Anfangswinkel>,<Endwinkel>
Zeichnet eine Ellipse oder ein Ellipsensegment.

PI
PI=<Variable>
Enthält den Wert von pi.

POKE
POKE_B(<Adresse>,<Daten>)
Schreibt einen 8-Bit Wert an die Speicheradresse.

POKE(<Adresse>,<Daten>)
Schreibt einen 16-Bit Wert an die Speicheradresse.

POKE_L(<Adresse>,<Daten>)
Schreibt einen 32-Bit Wert an die Speicheradresse.

POS
POS(<Dateinummer>)
Gibt die Zahl der gedruckten Zeichen seit der letzten neuen Zeile aus, oder die gegenwärtige Position des Cursors auf dem Bildschirm oder dem Drucker.

PRINT
PRINT[<Druckdaten>]<;,>[<Druckdaten>[<;,>]...]]
?[<Druckdaten>]<;,>[<Druckdaten>[<;,>]...]]
Zeigt Daten auf dem Bildschirm an.

PRINT USING
PRINT USING<"Formatstring">;<Variablenliste>
Zeigt Daten auf dem Bildschirm an unter Verwendung des angegebenen Formats.

PRINT #<Dateinummer>,USING<"Formatstring">;
<Ausdruck>[,<Ausdruck>..]
Schreibt Daten in eine Datei unter Verwendung des angegebenen Formats.

PRINT #
PRINT #<Dateinummer>,<Druckdaten>
[<Druckdaten>...]
Gibt Daten in eine Datei aus.

PTSIN
PTSIN(<Offset>)=(Ausdruck)
Eine VDI-Systemvariable, die als Array im Programm benutzt werden kann.

PTSOUT
PTSOUT(<Offset>)=(Ausdruck)
Eine VDI-Systemvariable, die als Array im Programm benutzt werden kann.

PUT
PUT[#]<Dateinummer>[,<Satznummer>]
Schreibt einen Satz in eine Randomdatei.

QUIT
QUIT
Verläßt ST BASIC und kehrt zum GEM Desktop zurück.

RANDOMIZE
RANDOMIZE[<Numerischer Ausdruck>]
Ruft den Zufallszahlengenerator auf.

READ
READ<Variable>,<Variable>...
Weist Daten aus dem DATA-Befehl der (den) angegebenen Variablen zu.

REM	REM<Kommentar> '<Kommentar> Ermöglicht das Einfügen von Kommentaren in das Programm.
RENUM	RENUM[<Neue erste Zeile>][,<Erhöhung>] Nummeriert die Zeilen des gegenwärtigen Programms neu.
REPLACE	REPLACE[<Dateiname>][,<Zeilennummernliste>] Ersetzt ein bestehendes Programm durch eine neue Version.
RESET	RESET Speichert den Inhalt des Fensters "Output" in den Graphikpuffer.
RESTORE	RESTORE[<Zeilenbezeichnung>] Setzt den Zeiger auf den angegebenen DATA-Befehl.
RESUME	RESUME[NEXT0<Zeilenbezeichnung>] Definiert den Rücksprungpunkt nach einer Fehlerabfangroutine.
RETURN	RETURN Markiert das Ende eines Unterprogramms.
RGB	RGB<Reg>,<r>,<g>,<b> Weist der Farbpalette die angegebenen Rot-, Grün- und Blauanteile zu.
RIGHT$	RIGHT$(<Zielstring>,<Integerzahl>) Gibt einen Teilstring vom Ende des Zielstrings aus.
RND	RND[(<Numerischer Ausdruck>)] Gibt eine Zufallszahl aus.
RSET	RSET<Stringvariable>=<Stringausdruck> Bringt die Daten rechts ausgerichtet in die Stringvariable.
RUN	RUN[<Dateiname>][,<Zeilenbezeichnung>] Führt das Programm aus.
SAVE	SAVE[<Dateiname>][,<Zeilenbezeichnung>] Speichert das gegenwärtige Programm im Quellenformat.
SGN	SGN[(<Numerischer Ausdruck>)] Gibt das Vorzeichen einer Zahl aus.

SIN SIN(<Numerischer Ausdruck>)
Gibt den Sinus des Arguments aus.

SOUND SOUND<Stimme>,<Lautstärke>,<Note>,
<Oktave>,<Länge>
Spielt Musiknoten.

SPACE$ SPACE$(<Numerischer Ausdruck>)
Gibt einen String der angegebenen Länge aus, gefüllt mit Leerzeichen.

SPC PRINT SPC(<Numerischer Ausdruck>)
Fügt die angegebene Zahl von Leerzeichen in den Druckstring ein.

SQR SQR(<Numerischer Ausdruck>)
Gibt die Quadratwurzel des Arguments aus.

SSHAPE SSHAPE<x1,y1>;<x2,y2>,<Array>
Sichert ein Raster in das angegebene Array.

STATUS STATUS=<Variable>
Enthält den Wert, der von jedem Aufruf von TOS™, GEM, VDI oder AES zurückgegeben wird.

STEP STEP[<Dateiname>][,<Zeilenbezeichnungs-nummer>]
Führt das Programm Zeile für Zeile aus.

STOP STOP
Hält die Programmausführung an.

STR$ STR$(<Numerischer Ausdruck>)
Wandelt das numerische Argument in einen String um.

STRING$ STRING$(<Numerischer Ausdruck>, <Numerischer Ausdruck><Stringausdruck>)
Gibt einen String der angegebenen Länge aus, gefüllt mit dem spezifizierten Zeichen.

SWAP SWAP<Erste Variable>,<Zweite Variable>
Tauscht die Werte der angegebenen Variablen aus.

SYSTAB SYSTAB(<Offset>)=<Ausdruck>
Ermöglicht den Zugriff auf die Systemzeigertabelle.

SYSTEM SYSTEM
Verläßt ST BASIC und kehrt zum GEM Desktop zurück.

TAB PRINTTAB(<Tabulatorposition>)
Fügt in einen PRINT-Befehl Tabulatoren ein.

TAN TAN(<Winkel im Bogenmaß>)
Gibt den Tangens des Arguments aus.

TRACE TRACE[<Zeilennummer>-<Zeilennummer>]
Führt das Programm aus und druckt dabei die angegebenen Zeilen.

TROFF TROFF[<Zeilennummer>-<Zeilennummer>]
Schaltet TRON aus.

TRON TRON[<Zeilennummer>-<Zeilennummer>]
Führt das Programm aus und druckt dabei die angegebenen Zeilen und Variablenwerte.

UNBREAK UNBREAK[<Zeilennummer>-<Zeilennummer>]
Schaltet BREAK aus.

UNFOLLOW UNFOLLOW[<Variable>[,<Variable>]...]
Schaltet FOLLOW aus.

UNTRACE UNTRACE[<Zeilennummer>-<Zeilennummer>]
Schaltet TRACE aus.

VAL VAL(<Stringausdruck>)
Gibt den numerischen Wert des angegebenen String aus.

VARPTR VARPTR(<Variable> #<Dateinummer>)
Gibt den Offset des Parameters vom "heap-segment" aus.

VDISYS VDISYS[(Scheinargument)]
Ermöglicht dem Benutzer den Zugriff auf die VDI-Schnittstelle des Betriebssystems.

WAVE WAVE<Schalter>,<Hüllkurve>,<Form>,
<Periode>,<Verzögerung>
Bestimmt die Gestalt der Wellenform, die in SOUND-Befehlen benutzt wird.

WEND WEND
Bestimmt das Ende zu einem WHILE.

WHILE WHILE<Logischer Ausdruck>
Definiert den Anfang und die Bedingung einer Endlosschleife.

WIDTH	WIDTH[#<Dateinummer>,]<Breite> Bestimmt die Breite der Bildschirmausgabe.
	WIDTH LPRINT<Breite> Bestimmt die Breite der Druckerausgabe.
WRITE	WRITE[<Ausdruck>][,<Ausdruck>] Zeigt die Ausgabe auf dem Bildschirm an.
WRITE#	WRITE#<Dateinummer>,<Ausdruck>[,<Ausdruck>]... Gibt Daten in eine sequentielle Datei aus.
XBIOS	XBIOS<Funktion>[,<Argumentenliste>] Ermöglicht einen Betriebssystemaufruf des XBIOS.

Anmerkung: <Beziehung> ist ein numerischer Ausdruck, dessen Ergebnis eine Integerzahl ist. Wenn der Wert Null ist, gilt die Beziehung als falsch, wenn er nicht Null ist, als wahr.

Sind Sie an der Programmierung mit ST BASIC interessiert? Dann ist das neue ST BASIC Handbuch das Hilfsmittel, das Sie brauchen.

Dieses über 300 Seiten lange Handbuch schließt einen kompletten Lehrgang für den beginnenden Programmierer sowie eine ausführliche Referenzsektion für den erfahrenen BASIC-Programmierer ein. Das ST BASIC Handbuch ist die einzige Unterlage, die Sie benötigen, um das neue BASIC voll auszunutzen.

Wegen weiterer Informationen fragen Sie bitte Ihren Atarihändler nach dem ST BASIC Handbuch .



C100536-004
Printed in Taiwan
K. I. 2. 1988